Project 2: Final Writeup

Bartolo Medical Sales Inc. Customer Relationship Management Database

Michelle Bartolo
Software Development- CMPT 220L
Professor Pablo Rivas
6 April 2016

Abstract

This paper details the motivations, process, and methods to create a database for Bartolo Medical Sales Inc. I wanted to create a database in order to organize the data for my father's business, Bartolo Medical Sales, in order to help him organize information about the companies he represents and customers he sells to. I utilized the Netbeans, which is an open-source integrated development environment (IDE) for developing with Java and other programming languages, to create the database that connects to a Java Darby or Java DB database. For my project, I created a database with a user interface that shows information for customers that buy from Bartolo Medical Sales. For each customer entered into the database, the name of the company, the name of the contact, their phone number, email, and address are stored into a table.

Introduction

Bartolo Medical Sales is a small Hospital and Health Care company founded in January 1991 by Domenic Bartolo, a graduate of Bernard Baruch College. The company sells various medical products in the Metropolitan New York area to a diversified medical dealer network. Bartolo Medical Sales evaluates the needs of prospective clients, drafts proposals, and makes formal presentations and demonstrations to market products and close sales. I was motivated to complete this project of creating a Customer Relationship Management (CRM) Program for the company in order to improve the sales of Bartolo Medical Sales and allow it to become increasingly successful. It is extremely beneficial for a business to use some sort of CRM program because it enables individuals to keep track of communications with customers in an organized manner. Unfortunately, these programs cost hundreds and sometimes thousands of dollars and do not always have all of the features wanted and needed. This database stores

information, such as email, phone number, and address, of clients so that Bartolo Medical Sales Inc. can continue to effectively sell medical equipment without having to pay thousands of dollars purchasing programs.

Detailed System Description

Java uses JDBC or Java Database Connectivity to connect to databases. There's a JDBC API, which is the programming part, and a JDBC Driver Manager, which programs use to connect to the database. JDBC allows a connection to various types of databases, such as Oracle, MySQL, etc. I used the inbuilt database that you get with the Java/NetBeans software. The database is called Java DB, which is a version of Apache Derby. It runs on a virtual server, which stops and starts from within NetBeans. The first thing I did was start a server at JavaDB, which allowed me to create a database. I then created a table for the database with columns called “ID”, “Company_Name,” “Contact_Name”, “Phone_Number”, “Email”, and “Address.” The ID column holds a unique identifying number, which identifies a specific row in the table. It is a primary key, which means it cannot be null and it is an integer. For the rest of the columns, I set the datatype as varchar. I then added data to this table by using sql commands to add new rows.

After I created the database, I started a new Java Application project and called the package database_console, and the main class DBConnect. To connect to a database, I had to add imports for connection, driver manager and sql exception. To set up a connection to a database, I added a line of code that takes my username, password, and host location of the database. I put that into a try...catch statement to make sure that the SQLException error doesn’t occur and

added a client driver which allows it to be connected successfully. Using various commands that connect the columns of the table I created to the code, I printed out the rows of the table.

After that, I created a user interface for the database, which allows the users to see their records in a more organized manner. It lets the user press buttons to scroll backwards and forwards through their data, go to the first and last records, add new records, update a record, delete a record, save the record they want to add

or record if they change their mind. To do this I made a JFrame form which allowed me to create a GUI application and connect it to my database using java.

To connect the "Next" button I created with the GUI application to my code and make it functional, I created an action performed method which connects with the variable name of the button. I used an "if, else" statement check to check if there is a next record to move to, rs.next() and if there is a next record, I made it display it in the Text Fields. The if else statement is wrapped in a try … catch block, to prevent a SQLException error. If there's no next record, an alert pops up saying "End of file."

To connect the "Previous" button that I created with the GUI application to my code and to make it functional, I created an action performed method which connects with the variable name of the button. To go backwards through the records, I utilized the same commands as the next button, but instead of rs.next() I used the rs.previous() function.

To connect the "First" button I created with the GUI application to my code and make it functional, I created an action performed method which connects with the variable name of the button. I did not need an "if, else" statement because it is not necessary to check if there is a next or previous record since we are just going to the first record. The only thing I needed to do was

move to the first record with rs.First, then display the first record in the Text Fields. To move to the last record using the last record button, I used the same commands as I did for the first record button, but I used rs.Last instead of rs.First.

To update an existing record, I created an action performed method that connects to the variable name of the button. The first thing to do is get the text from the Text Fields using .getText() and to update an ID field, I had need to convert the String to an Integer using ParseInt(). Now that I have all the data from the Text Fields, I called the update methods of the ResultSet object, which rs.updateString(and inside of it I put the name of column and name of the string we just got the text from. To commit the changes to the database, you issue an updateRow command: rs.updateRow( ). Everything is enclosed in a try catch block again, in case there's a SQLException error.

To add a new record, I created an action performed method that connects to the variable name of the button. When the new record button is clicked, all other buttons except for Save new record and cancel new record are disabled. To get which row that is currently being pointed to in the Result Set there is a method called getRow. This allows the row number you are at to be stored in the result set. Also when the user, clicks this button, all the forms are cleared, but setting the text in each text field to an empty string. Again, everything is wrapped in a try catch block.

In order to implement the cancel button, I had to get the row that was previously loaded and put the data back in the Text Fields. To move back to the row that was previously being pointed to, I used the absolute method: rs.absolute(curRow). The absolute method moves to a

fixed position in the ResultSet. It is moved to the value that we stored in the variable curRow. Once the correct row is being pointed at, the data can be loaded into the Text Fields.

Before it Is possible to save a new record, I had to move to the Insert Row to create a blank record in the ResultSet. I then added the data to the ResultSet with rs.moveToInsertRow( ) and updated each string by using name of column and the name of the string and then rs.insertRow(), which after adding the data to the ResultSet, inserts a new row. To commit any changes to the database I had to close the Statement object and ResultSet object. This allows all of the records to be reloaded with the new data remains.

To delete a record, I used the deleteRow method of the ResultSet, which is rs.deleteRow(); However, the Driver I used, the ClientDriver, leaves a blank row in place of the data that was deleted. Because of this, if you try to move to that row using the Next or Previous buttons, the ID Text Field will have a 0 in it, and all the others will be blank. To solve this problem I first deleted a row then, again like before, close the Statement object and the ResultSet objects. We can then reload all the data in the Text Fields and there we won't be any blank rows.

Requirements

In order to run this program, it is necessary to have a computer, Java installed on the computer, and Netbeans software.

Literature Survey

In the past, Bartolo Medical Sales Inc. has purchased Customer Relationship Management Programs, such as Goldmine or ACT Software. Unfortunately, these programs cost hundreds and sometimes thousands of dollars and do not always have all of the features wanted and needed. Most recently, upon attempting to utilize Goldmine when Windows 10 was

introduced, the system would not work and crashed. My database project can be customized to fit the needs of Bartolo Medical Sales and does not cost the company any money. For example, right now, I have columns to store information about company name, customer name, phone number, email, and address; however, in the future, I can easily update it to include more information. Since I made this database, Bartolo Medical Sales, can communicate with me personally for these changes, rather than speaking to a big company that may not respond.

User Manual

In order to utilize this system, the user must run the program from Netbeans. I created a user interface using the Java GUI Swing Application to make it extremely easy to navigate and understand. Each column of the database (ID, Company name, Customer name, phone number, email, and address) are displayed as text fields with labels next to them. The user can click the "Next" button to move forward through the information in the database. They can click the "Previous" button to move backwards through the information in the database. The "First" button brings the user to the first record and the "Last" button brings the user to the last record. The "Update" record button allows the user to alter an existing record. The "Delete Record" buttons removes the selected record from the database. Upon clicking"Add New Record" button, all of the text fields are cleared. The user can then type in a new record (making sure not to repeat any numbers for the ID because that would result in an error) and click the "Save New Record" button to store the information in the database. The "Cancel New Record" reloads the record from before the "Add New Record" button was clicked.

Conclusion

The database application I created for Bartolo Medical Sales is extremely significant because it has a real life application. Many projects created for various classes are not used again after they are submitted; however, this project will help a real company organize real customer information. Bartolo Medical Sales will benefit from the creation of this database because they will no longer have to purchase software to store their customer information into a database. In the future, I will continue updating this database to fit the needs of Bartolo Medical Sales. I will be in contact with Domenico Bartolo, the president with Bartolo Medical Sales in order to customize features of the application to fit the company's needs and ensure the most effective usage.

References

LIANG, Y. DANIEL. *INTRODUCTION TO JAVA PROGRAMMING*. S.l.: PRENTICE HALL, 2017. Print.

"NetBeans IDE." *Welcome to NetBeans*. N.p., n.d. Web. 06 May 2017. <https://netbeans.org/>.

"Trail: Collections: Table of Contents." *Trail: Collections: Table of Contents (The Java™ Tutorials)*. N.p., n.d. Web. 06 May 2017. <http://docs.oracle.com/javase/tutorial/collections/TOC.html>.